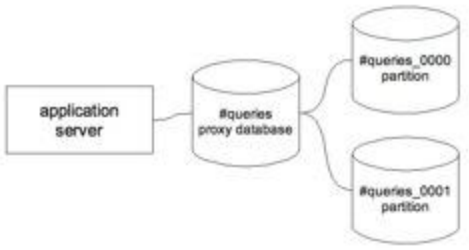
# PostgreSQL 数据库集群和 PL/Proxy 配置安装指南

PL/Proxy 和 PostgreSQL 集群的结构关系可以用下图清楚地表示，对 PL/Proxy 和 PostgreSQL 集群还不太了解的朋友可以看 Skype Plans for PostgreSQL to Scale to 1 Billion Users 这篇文章。




以下操作是在三台不同机器上执行的情况，其中 plproxy 节点的机器名是 PLPROXY，数据库节点的机器名分别是 database1 和 database2。机器硬件配置如下，同时需要 Linux-4.2、postgresql-8.3.4 和 plproxy-2.0.4，pgbouncer 的安装过程略去。

plproxy 节点和数据库节点机器配置及环境

plproxy 节点：

hostname：plproxy

IP address：192.168.1.193

OS：openSuSE Enterprise Linux 11

CPU：Intel(R) Pentium(R) Dual E2180 @ 2.00GHz

MemTotal：512M

node1 节点：

hostname：database1

IP address：192.168.1.172

OS：openSuSE Enterprise Linux 11

CPU：Intel(R) Pentium(R) Dual E2180 @ 2.00GHz

MemTotal：256M

node2 节点：

hostname：database2

IP address：192.168.1.175

OS：openSuSE Enterprise Linux 11

CPU：Intel(R) Pentium(R) Dual E2180 @ 2.00GHz

MemTotal：256M

1. 在 plproxy,database1,database2 上安装 postgresql-8.3.4，并创建 URTCluster 数据库

```
## Compile and install
#gunzip postgresql-8.3.4.tar.gz
tar xf postgresql-8.3.4.tar
#cd postgresql-8.3.4
#./configure –prefix=/home/y/pgsql –with-perl          //"y"是用户家目录
#gmake
#gmake check
#sudo gmake install
```

## Add Linux System User

#sudo /usr/sbin/useradd –m postgres //建立 postgres 用户

#passwd postgres //设置 postgres 用户密码

##Create Database

sudo mkdir /home/y/pgsql/dbdata //建立数据库文件夹，”y”是用户家目录

sudo chown postgres /home/y/pgsql/dbdata ///home/y/pgsql/dbdata 文件夹所有者修改为 postgres，”y”是用户家目录

## Init database and Start service

```
fixing permissions on existing directory /home/database2/pgsql/dbdata ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers/max_fsm_pages ... 32MB/204800
creating configuration files ... ok
creating template1 database in /home/database2/pgsql/dbdata/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    /home/database2/pgsql/bin/postgres -D /home/database2/pgsql/dbdata
or
    /home/database2/pgsql/bin/pg_ctl -D /home/database2/pgsql/dbdata -l logfile start

postgres@database2:~>
```

sudo -u postgres /home/y/pgsql/bin/initdb -D /home/y/pgsql/dbdata //初始化数据库，”y”是用户家目录

```
postgres@database2:~> /home/database2/pgsql/bin/pg_ctl start -D /home/database2/pgsql/dbdata/ -m
 fast &
[1] 21657
postgres@database2:~> server starting
LOG:  database system was shut down at 2008-11-06 17:12:11 CST
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections

[1]+  Done                    /home/database2/pgsql/bin/pg_ctl start -D /home/database2/pgsql/db
data/ -m fast
postgres@database2:~>
```

sudo -u postgres /home/y/pgsql/bin/pg_ctl start -D /home/y/pgsql/dbdata –m fast & //启动 postgresql 数据库，”y”是用户家目录

## Create DB and Use Local Connection

```
postgres@database2:~> /home/database2/pgsql/bin/createdb URTCluster
postgres@database2:~> /home/database2/pgsql/bin/psql -d URTCluster
Welcome to psql 8.3.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

URTCluster=#
```

sudo -u postgres /home/y/pgsql/bin/createdb URTCluster //建立 URTCluster 数据库"y"是用户家目录

##检查数据库是否已经创建

sudo -u postgres /home/y/pgsql/bin/psql -d URTCluster //"y"是用户家目录

#database1,database2 必须允许 plproxy 访问

#编辑 postgresql.conf,打开 tcp 连接端口

sudo vim /home/y/pgsql/dbdata/postgresql.conf

listen_addresses = '*' //监听所有地址

port = 5432 //设置 postgresql 端口为 5432

```
# - Connection Settings -

listen_addresses = '*'
#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost', '*' = all
                                        # (change requires restart)
port = 5432
#port = 5432                            # (change requires restart)
max_connections = 100                   # (change requires restart)
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).  You might
# also need to raise shared_buffers to support more connections.
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directory = ''             # (change requires restart)
#unix_socket_group = ''                 # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
                                        # (change requires restart)
#bonjour_name = ''                      # defaults to the computer name
                                        # (change requires restart)
```

#添加 postgres 用户的认证

#sudo vim /home/y/pgsql/dbdata/pg_hba.conf

host URTCluster postgres 192.168.1.0/24 trust

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               trust
# IPv4 local connections:
host    all         all         127.0.0.1/32          trust
host    URTCluster      postgres        192.168.1.0/24          trust
# IPv6 local connections:
```

# 重起服务器

#sudo -u postgres /home/y/pgsql/bin/pg_ctl -D /home/y/pgsql/dbdata stop

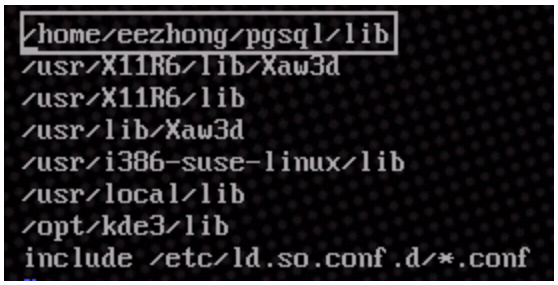
#sudo -u postgres /home/y/pgsql/bin/pg_ctl start -D /home/y/pgsql/dbdata –m fast &　　//"y"是用户家目录

#sudo -u postgres /home/y/pgsql/bin/pg_ctl -D /home/y/pgsql/dbdata reload

```
postgres@database2:~> /home/database2/pgsql/bin/pg_ctl -D /home/database2/pgsql/dbdata stop
waiting for server to shut down....LOG:  received smart shutdown request
LOG:  autovacuum launcher shutting down
LOG:  shutting down
LOG:  database system is shut down
 done
server stopped
postgres@database2:~> /home/database2/pgsql/bin/pg_ctl start -D /home/database2/pgsql/dbdata -m
fast &
[1] 21721
postgres@database2:~> server starting
LOG:  could not create IPv6 socket: Address family not supported by protocol
LOG:  database system was shut down at 2008-11-06 17:24:55 CST
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections

[1]+  Done                    /home/database2/pgsql/bin/pg_ctl start -D /home/database2/pgsql/db
data -m fast
postgres@database2:~> /home/database2/pgsql/bin/pg_ctl -D /home/database2/pgsql/dbdata reload
LOG:  received SIGHUP, reloading configuration files
server signaled
postgres@database2:~> _
```

#vi /etc/ld.so.conf　　编辑/etc/ld.so.conf 文件，添加/home/y/pgsql/lib，把 postgresql 的 lib 进行加载




#ldconfig　　//重新加载系统 lib 路径

2. 在 plproxy 上安装 plproxy-2.0.4（安装 plproxy 必须使用 root 用户，否则不能正常安装）

#检查$PATH 变量里是否有/home/y/pgsql/bin 目录，如果没有，修改你的.bash_profile 文件，添加 /home/y/pgsql/bin 到 path 里。（root 用户：/etc/profile；普通用户：~/.profile；postgres 用户：.bash_profile）

在文件底部添加：

PATH=$PATH:/home/y/pgsql/bin

echo PATH

#soure /usr/lib/postgres/.bash_profile

#source /etc/profile

#source /home/y/.profile

echo $PATH

#gunzip plproxy-2.0.4.tar.gz

#tar xf plproxy-2.0.4.tar

#cd plproxy-2.0.4

#gmake

#sudo gmake install

plproxy 安装错误检查步骤

1、检查 home/y/pgsql/share/contrib/plproxy.sql 文件是否存在；

2、查看 root、postgres、"y"用户$PATH 变量是否有/home/y/pgsql/bin 目录，如果没有，修改你的.bash_profile 文件底部添加$PATH 变量/home/y/pgsql/bin 目录；

3、删除 plproxy 解压缩代码重新安装；

4、编译的时候用户是否有权限安装，plproxy 默认只能由 root 用户编译安装；

#创建 plproxy

#sudo -u postgres /home/y/pgsql/bin/psql –f /home/y/pgsql/share/contrib/plproxy.sql URTCluster

倒入 plproxy.sql 文件错误检查步骤：

1、检查 plproxy 是否安装成功；

2、当前用户是是为 postgres 用户（#whoami）；

3、查看系统是否对 postgresql 的 lib 路径进行加载（#cat /etc/ld.so.conf）；

4、postgresql 服务是否正常启动，查看进程（#ps –eaf | grep postgres）；

5、查看建立和初始化数据库是否有错误；

6、查看 URTCluster 数据库是否成功建立（#sudo -u postgres /home/y/pgsql/bin/psql -d URTCluster URTCluster=#）；

7、查看硬盘是否有空间（#df）

3. 在 plproxy,database1,database2 上安装 plpgsql

#sudo -u postgres /home/y/pgsql/bin/createlang plpgsql URTCluster

```
postgres@suselinux:~> /home/eezhong/pgsql/bin/createlang plpgsql URTCluster
postgres@suselinux:~>
```

4. 在 plproxy 上创建 schema

#sudo -u postgres /home/y/pgsql/bin/psql -d URTCluster

URTCluster=# create schema plproxy;

5. 在 plproxy 上初始化设置

#plproxy 的配置是通过三个函数（过程）实现的，这三个函数的标准模版如下：

#这个函数是让 plproxy 可以找到对应的集群

CREATE OR REPLACE FUNCTION plproxy.get_cluster_partitions(cluster_name text)

//get_cluster_partitions 是函数名称

RETURNS SETOF text AS $$ //SETOF 是返回多条记录，text 使数据类型

BEGIN

IF cluster_name ='eezhong' THEN //cluster_name 是群集的名字

RETURN NEXT 'dbname=URTCluster host=192.168.1.172'; //数据库节点的数据库名和 IP 地址

RETURN NEXT 'dbname=URTCluster host=192.168.1.175'; //数据库节点的数据库名和 IP 地址

RETURN;

END IF;

RAISE EXCEPTION 'Unknown cluster'; //如果群集名不存在，抛出异常，这个是在数据库内部处理的，最终会写入日志中。'Unknown cluster'是报错信息

END;

$$ LANGUAGE plpgsql;

#这个函数是 plproxy 用于判断是否给前端返回已经 cache 过的结果用的

```
CREATE OR REPLACE FUNCTION plproxy.get_cluster_version(cluster_name text)
// get_cluster_version 是函数名称
RETURNS int4 AS $$
BEGIN
IF cluster_name = 'eezhong' THEN
RETURN 1;
END IF;
RAISE EXCEPTION 'Unknown cluster';
END;
$$ LANGUAGE plpgsql;
```

#这个函数是获取不同的集群的配置

```
create or replace function plproxy.get_cluster_config(cluster_name text, out key text, out val text)
// get_cluster_config 是函数名称；text 表示变量类型，out 表示是一个输出参数；out key text 表示声明一个输出参数，变量名为 key，类型 text；out val text 同上
returns setof record as $$
begin
key := 'statement_timeout';          //就是给 key 变量赋值，赋的值为'statement_timeout'
val := 60;          //就是给 key 变量赋值，赋的值为 60
return next;
return;
end;
$$ language plpgsql;
```

#把这三个函数放在一个 URTClusterInit.sql 文件里，并执行

#sudo -u postgres /home/y/pgsql/bin/psql -f URTClusterInit.sql -d URTCluster -h 192.168.1.193

```
postgres@suselinux:~> /home/eezhong/pgsql/bin/psql -f /home/eezhong/pgsql/URTClusterInit.sql -d
URTCluster -h 192.168.1.193
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
postgres@suselinux:~> _
```

6. 在 database1，database2 节点上设置（两个节点信息必须一致）

#给每个数据库节点都创建一张表 users

```
CREATE TABLE china (
username text,
email text
);
```

#给每个数据库节点都创建一个插入函数

```
CREATE OR REPLACE FUNCTION insert_user(i_username text, i_emailaddress text)
RETURNS integer AS $$          //函数的返回值是 integer 类型
INSERT INTO china (username, email) VALUES ($1,$2);
SELECT 1;
$$ LANGUAGE SQL;
```

```
--create tables users
CREATE TABLE china(
username text,
email text
);
--create insert function
CREATE OR REPLACE FUNCTION insert_user(i_username text,i_email text)
RETURNS integer AS $$
INSERT INTO china(username,email) VALUES ($1,$2);
SELECT 1;
$$ LANGUAGE SQL;
```

#把函数保存在 URTClusterNodesInit_1.sql 文件里，并执行

sudo -u postgres /home/y/pgsql/bin/psql -f URTClusterNodesInit_1.sql -h 192.168.1.172 -d URTCluster

sudo -u postgres /home/y/pgsql/bin/psql -f URTClusterNodesInit_1.sql -h 192.168.1.175 -d URTCluster

```
postgres@database1:~> /home/database1/pgsql/bin/psql -f /home/database1/pgsql/URTClusterNodesInit_1.
sql -h 192.168.1.172 -d URTCluster
CREATE TABLE
CREATE FUNCTION
postgres@database1:~>
```

```
postgres@database2:~> /home/database2/pgsql/bin/psql -f /home/database2/pgsql/URTClusterNodesInit_1.
sql -h 192.168.1.175 -d URTCluster
CREATE TABLE
CREATE FUNCTION
postgres@database2:~>
```

7. 在 plroxy 节点上设置

#在 plproxy 节点上创建一个同名的插入函数，用于进行集群检索

CREATE OR REPLACE FUNCTION insert_user(i_username text, i_emailaddress text)

RETURNS integer AS $$ //函数的返回值是 integer 类型

CLUSTER ‘eezhong’;

RUN ON hashtext(i_username);

$$ LANGUAGE plproxy;

#在 plproxy 节点上创建一个查询函数，用于进行集群检索

CREATE OR REPLACE FUNCTION get_user_email(i_username text)

RETURNS text AS $$ ////函数的返回值是 text 类型

CLUSTER ‘eezhong’; //集群的名字

RUN ON hashtext(i_username) ;

SELECT email FROM users WHERE username = i_username;

$$ LANGUAGE plproxy;

```
--/cluster
CREATE OR REPLACE FUNCTION insert_user(i_username text,i_emailaddress text)
RETURNS integer AS $$
CLUSTER 'eezhong';
RUN ON hashtext(i_username);
$$ LANGUAGE plproxy;

--cluster
CREATE OR REPLACE FUNCTION get_user_email(i_username text)
RETURNS text AS $$
CLUSTER 'eezhong';
RUN ON hashtext(i_username);
SELECT email FROM china WHERE username = i_username;
$$ LANGUAGE plproxy;
```

#把函数保存在 URTClusterProxyExec.sql 文件里，并执行

sudo -u postgres /home/y/pgsql/bin/psql -f URTClusterProxyExec_1.sql -h 10.0.0.1 -d URTCluster

```
postgres@suselinux:~> /home/eezhong/pgsql/bin/psql -f /home/eezhong/pgsql/URTClusterProxyExec_1.
sql -h 192.168.1.193 -d URTCluster
CREATE FUNCTION
CREATE FUNCTION
postgres@suselinux:~>
```

8. 在 plproxy 上测试结果

sudo -u postgres /home/y/pgsql/bin/psql -d URTCluster

SELECT insert_user('Sven','sven@somewhere.com');

#被保存到 database2， 可以用 select hashtext('Sven') & 1 验证，被 hash 到 partition 1

SELECT insert_user('Marko', 'marko@somewhere.com');

#被保存到 DATABASE2， 可以用 select hashtext('Marko') & 1 验证，被 hash 到 partition 1

SELECT insert_user('Steve','steve@somewhere.cm');

#被保存到 database1， 可以用 select hashtext('Steve') & 1 验证，，被 hash 到 partition 0

SELECT get_user_email('Sven');

SELECT get_user_email('Marko');

SELECT get_user_email('Steve');